Nicholas Bowman, Sonja Johnson-Yu, Kylie Jue **Section #7**
CS106AP **August 6, 2019**

Section #7: Solutions

1. Sorting with `lambdas`

```
>>> strs = ['apple', 'BANANA', 'candy', 'aardvark']
# Solution for Part 1
>>> sorted(strs, key=lambda word: word.lower())
['aardvark', 'apple', 'BANANA', 'candy']
# Solution for Part 2
>>> sorted(strs, key=lambda word: word.lower()[len(word)-1])
['BANANA', 'apple', 'aardvark', 'candy']

# Solution for Part 3
>>> nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> sorted(nums, key=lambda num: abs(num-3.14))
[3, 4, 2, 5, 1, 6, 7, 8, 9, 10]
>>> min(nums, key=lambda num: abs(num-3.14))
3
>>> max(nums, key=lambda num:abs(num-3.14))
10

>>> rentals = [('main st.', 4, 4000),('elm st.', 1, 1200),('pine st.', 2, 1600)]
# Solution for Part 4a
>>> sorted(rentals, key=lambda room_tuple: room_tuple[1])
[('elm st.', 1, 1200), ('pine st.', 2, 1600), ('main st.', 4, 4000)]
# Solution for Part 4b
>>> sorted(rentals, key=lambda room_tuple: room_tuple[2])
[('elm st.', 1, 1200), ('pine st.', 2, 1600), ('main st.', 4, 4000)]
# Solution for Part 4c
>>> sorted(rentals, key=lambda room_tuple: room_tuple[2]/room_tuple[1])
[('pine st.', 2, 1600), ('main st.', 4, 4000), ('elm st.', 1, 1200)]
>>> min(rentals, key=lambda room_tuple:room_tuple[2]/room_tuple[1])
('pine st.', 2, 1600)
```

2. One Liners

Comprehensions

```
>>> stats = [('foo.com', 'firefox', 23), ('bar.com', 'chrome', 47), ('foo.com',
'chrome', 3), ('bar.com', 'firefox',
16)]
>>>
# Solution for Part 1
>>> [tuple[2] for tuple in stats if tuple[0] == 'foo.com']
[23, 3]
# Solution for Part 2
>>> sum([tuple[2] for tuple in stats if tuple[0] == 'foo.com'])
26
# Solution for Part 3
>>> sum([tuple[2] for tuple in stats if tuple[1] == 'firefox'])
39
```

More List Comprehensions

```
>>> nums = range(20)
# Solution for Part 1
>>> [abs(num - 10) for num in nums]
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
# Solution for Part 2
>>> [abs(num - 10) for num in nums if num >= 10 and num <= 15]
[0, 1, 2, 3, 4, 5]
>>> pairs = [('zzz', 3), ('bbb', 10), ('ccc', 4), ('aaa', 6)]
# Solution for Part 3
>>> [pair[1] for pair in pairs]
[3, 10, 4, 6]
# Solution for Part 4
>>> [pair[1] for pair in pairs if not pair[0].startswith('c')]
[3, 10, 6]
# Solution for Part 5
>>> [pair[0][0].upper() + pair[0][1:] for pair in pairs]
['Zzz', 'Bbb', 'Ccc', 'Aaa']
```

3. Data Analysis Using Jupyter Notebooks

Download the solution Jupyter notebook (which you can run the same way as the starter code) here, or view a pdf of said notebook here.

4. Bonus: Experimenting with the Debugger!

There's no correct answer to this one! When it comes to using the debugger, the journey is the destination.